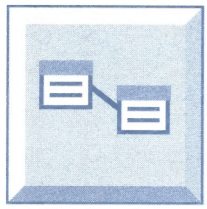
COBB

Inside MICROSOFT ACCESS

Tips & techniques for Microsoft Access • Windows

July 1993 • Vol. 1 No. 5



Query Tip

Selecting records based on summary information

When you want to place criteria in a query, you usually place the restrictions on the raw data in the table. However, you may occasionally want to base criteria on summary calculations. In other words, you may want to summarize the data and then select records according to how their data compares to the summary results. For instance, suppose you want to find the customers who have bought at least $1,000 worth of your product in the last year. Such a query would need to tally each customer's orders and then determine if each sum is greater than 1,000. In this article, we'll describe how to build queries that place this type of criteria.

IN THIS ISSUE

The technique

To base a query's criteria on a summary calculation, you simply open the Total row in the QBE grid, move to the column you want to summarize, and define the summary operation in the column's Total cell. Then, you move to the Criteria cell and enter the criteria on the summary results you want the records to meet.

Although our description of the technique seems simple, you'll usually run into a few complications. We'll describe these complications in a moment. Let's start instead with a simple example that demonstrates how useful this technique can be.

A simple example

Suppose you want to determine which states are the most profitable to do business in. One thing you might do is to see which states have the most customers. For instance, suppose you store customers' name and address data in a table named Customers. You can create a query that bases criteria on the count of customers from each state.

We'll borrow the Customers table from the NWIND sample database for this example, since it already contains plenty of data. Table A on page 2 shows the structure of this table. As you can see, it has Customer ID as its key field and includes the Region field to keep track of each customer's state (or province, for international customers).

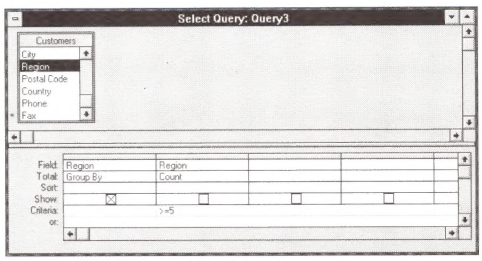
Let's now create a query that selects a Region entry only if there are at least five customers in that region. If you haven't already opened the NWIND database, do so now by using the File menu's Open Database... command. Then, highlight the Customers table in the Database window and

click the New Query button ( ) on the tool bar. Next, you drag the Region field from the table's field list to the QBE grid. If you ran the query at this point, the datasheet would then show all the regions in the Customers table. Now you want to place the criteria that tell Access to select only the regions with five or more customers.

You don't use the original Region column to set criteria. Doing so would place criteria on the table's raw data. In this situation, you want to place criteria on a summary result. To do so, you must use a second copy of the Region field in a separate column.

Drag the second copy of the field to the QBE grid and then deselect the Show cell's check box. Next, click the Totals button ( ) on the tool bar to open the Total row. Then, in the second column's Total cell, click the dropdown arrow and choose Count from the selection list. Finally, move to the Criteria cell and enter >=5. Figure A shows the query.

Figure A



This query selects Region entries only if the regions have at least five customers.

Table A

Key	Field Name	Data Type	Properties
	Customer ID	Text	Field Size=5
	Company Name	Text	Field Size=40
	Contact Name	Text	Field Size=30
	Contact Title	Text	Field Size=30
	Address	Text	Field Size=60
	City	Text	Field Size=15
	Region	Text	Field Size=15
	Postal Code	Text	Field Size=10
	Country	Text	Field Size=15
	Phone	Text	Field Size=24
	Fax	Text	Field Size=24

To see the results, click the Datasheet View button ( ) on the tool bar. Figure B shows the datasheet that identifies California, Oregon, and Washington as the only states that have at least five Northwind customers.

Before moving on, save the query by pulling down the File menu and selecting the Save Query option. In the Query Name dialog box, type the name *Regions With Five Customers* and click OK.

This example shows you how powerful this technique can be for extracting information from your data. However, as you create

Inside MICROSOFT ACCESS™

Inside Microsoft Access (ISSN 1067-8204) is published monthly by The Cobb Group.

Prices:
Domestic ..... $59/yr. ($7.00 each)
Outside US ..... $79/yr. ($8.50 each)

Phone:
Toll free ..... (800) 223-8720
Local ..... (502) 491-1900
Customer Relations Fax ..... (502) 491-8050
Editorial Department Fax ..... (502) 491-4200

Address:
You may address tips, special requests, and other correspondence to

The Editor, *Inside Microsoft Access*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Postmaster:
Second class postage is pending in Louisville, KY. Send address changes to

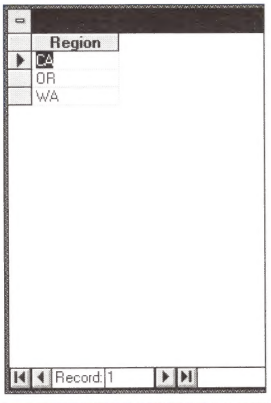
Inside Microsoft Access
P.O. Box 35160
Louisville, KY 40232

Staff:
Editor-in-Chief ..... David Brown
Editing ..... Timothy E. Hampton
Elizabeth Welch
Production Artist ..... Julie Jefferson
Design ..... Karl Feige
Publications Manager ..... Tara Dickerson
Managing Editor ..... Suzanne Thornberry
Circulation Manager ..... Brent Shean
Publications Director ..... Linda Baughman
Editorial Director ..... Jeff Yocom
Publishers ..... Mark Crane
Jon Pyles

Back Issues:
To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $7 each, $8.50 outside the US. You can pay with MasterCard, VISA, Discover, or American Express, or we can bill you. Please identify the issue you want by the month and year it was published. Customer Relations can also provide you with an issue-by-issue listing of all articles that have appeared in *Inside Microsoft Access*.

more complicated queries, implementing the technique can become a little tricky. For instance, you'll have a harder time displaying the names of all the customers that reside in the states the previous query selects. The problem stems from the way you tell the query which fields to summarize and which fields to display. To understand this complication, you need to understand the role of the Group By option.

Figure B



Only California, Oregon, and Washington have five or more Northwind customers.

The Group By option

The Group By option is one of many possible options in the Total cell, but all the others are types of summary operations. When you select Group By for a field, Access groups the data by that field before performing the summary calculations the other columns define. Furthermore, when you select Group By in more than one column, Access groups the data by the unique combinations of those fields.

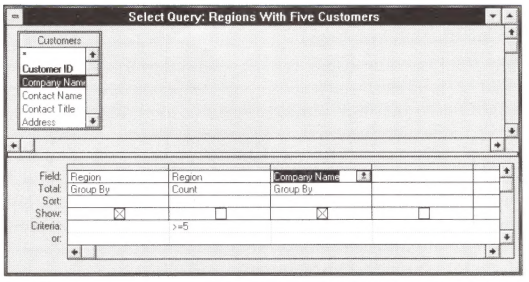
To see an example, return to the Regions With Five Customers query shown in Figure A. The Region column's Group By operator tells Access to group the Customer data by Region entries. Then, the Count summary operation in the second column can count the number of entries in each group.

How Group By affects criteria on summary results

Now, how does the Group By option complicate placing summary criteria? Well, Group By doesn't cause the problems by itself. The complicating issue here is that if you want to show a column in the datasheet, you must enter into the Total cell a summary operator such as Count or the Group By operator. Therefore, you can't include a field only for display purposes.

To demonstrate this property, we'll show you how to display the names of those customers that reside in the "high-traffic" regions that the Regions With Five Customers query selects. You first might try including the Company Name field in the QBE grid. However, when you do, you'll find you must include the Group By option in that column's Total cell, as shown in Figure C.

Figure C



As a first guess at the solution, you might simply try to include the Company Name field in the QBE grid.

When you run the query, Access will group records by unique combinations of Company Name and Region and then count the records in each group. Since the query doesn't group only by the Region field anymore, the query will no longer count the number of customers from each state. In fact, since Company Name is unique in this table, the query will never count more than 1 for any of the groups.

In short, you often can't directly apply summary criteria to a query that includes fields other than Group By and summary fields that define the summary calculation.

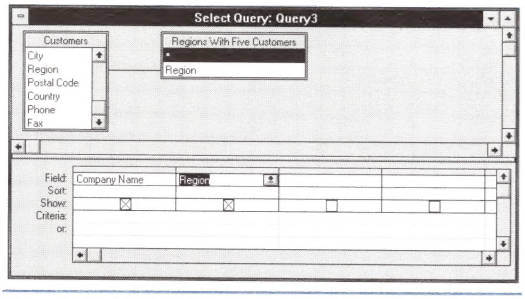
The full technique

To work around such problems, you must use two queries. The first query selects field entries according to how they compare to summary results, and the second query selects records that have entries the first query selects.

You'll see this technique more clearly in an example. Let's build on the query we created before that selects customers from regions that contain at least five customers. You already have available a query that can select the states. To get the names of the customers from those states, create a second query based on the first query and the Customers table.

To build this query, first build a basic query that selects the Company Name and Region from all records. Highlight the Customers table in the Database window and click the New Query button. Then, drag the Company Name and Region fields to the QBE grid.

Figure D



To select Company Names from the "high-traffic" states, you must build a second query that's based on the first.

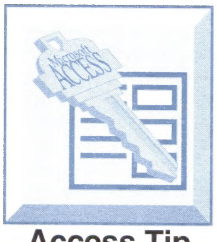
Next, you include the Regions With Five Customers query in this query and create an equi-join between the tables through the Region fields. To do so, return to the Database window by pressing [F11] and click the Query button to display the list of queries. Then, find the Regions With Five Customers query and drag it to the new query. When you do so, Access will place the query's field list in the new query's window. Then, return to that Query window and create the equi-join. Do so by clicking the Customers table's Region field and dragging it to the Regions With Five Customers query's Region field. Finally, save this query by using the File menu's Save As command, naming the query Popular States. Figure D shows the new query.

When you click the Datasheet View button to see the query results, Access will run both queries. Access will realize that before it can select customer names from the Customers table, it must run the Regions With Five Customers query to know which regions to select customer names from.

You'll usually need to use the double-query technique for basing criteria on summary information. You create a query to compute your summary results and to select records based on that result. Then you can draw on that query in the main query to select the data you actually want to see.

Conclusion

In this article, we showed how to base query criteria on summary calculations. With this technique, you can extract all sorts of interesting information from your data. Also, see our solution to Stephen Einson's letter beginning on page 14 for a novel use of this technique. ◆



Access Tip

Loading Access data into a Word for Windows data file

One advantage of working in the Windows environment is that applications work together. In this article, we'll show you how Access and Word for Windows cooperate to create a mail merge document's data file. If you've used Word for Windows' mail merge feature, you know you must create two files—the letter document you want to print and the data file containing the field data that changes from letter to letter. As you'll see, Microsoft has included features in Access that make it easy for you to create the data file.

Creating the data file

Loading data from Access into a Word for Windows data file is remarkably simple. You open a table or run a query, select the records in the datasheet you want to include in the data file, copy the records to the Clipboard, and then paste the records into an empty Word for Windows document.

The data will appear in the format Word for Windows requires for a mail merge data file—specifically, as tab-delimited data. Also, the first line will contain the field headers that Word for Windows needs in order to associate the columns of data with the appropriate mail merge fields.

To use this document as the data file of a mail merge document, you must save the document with a new filename and then attach the data file to your letter document by using the Print Merge command on the File menu. Once you attach the data file, you can reference its field in the text of your letter.

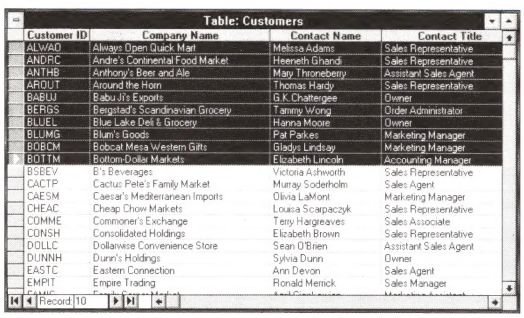
An example

Let's look at an example. Suppose you want to mail a thank-you letter to various customers. We'll use the Northwind database's Customers table that comes with Access as the source for the names and addresses for this letter.

You must first select the companies to whom you want to send the letter. Open the NWIND database and then double-click the Customers table in the Database window. In this simple exercise, you'll select just the first

ten records. To do this, click the tenth record's record selector (the tenth record has the Customer ID entry *BOTTM*) and then shift-click the first record's record selector. When you shift-click, Access will select all the records between the first and tenth, as shown in Figure A. Now that you've selected the first ten records, pull down the Edit menu and select Copy to copy the records to the Clipboard.

Figure A

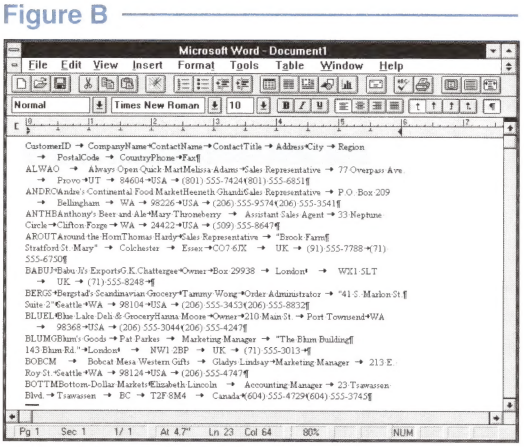


For this simple example, select and copy to the Clipboard the first ten records of the NWIND database's Customers table.

That's all you have to do in Access. The rest of the technique you perform in Word for Windows. Return to Windows' Program Manager and launch Word for Windows. The program will create a new document as it starts up. You just paste the Access records into it by pulling down Word for Windows' Edit menu and selecting Paste. The Access records will appear in the new document as shown in Figure B.

The data looks messy in Word for Windows because each record's paragraph wraps to a second line. However, if you look closely, you'll see the customer information is intact.

Figure B



You include the Access data in a Word for Windows document by simply pasting it from the Clipboard.

The first paragraph, which we've shaded in blue, holds the field headers. You'll notice the Access table's field names rather than data. The next ten paragraphs include the data from the ten records you copied to the Clipboard. Word for Windows knows to paste the data into this format automatically. Next, select the Save As... command on the File menu, enter *DATAFILE.DOC* in the File Name text box, and click OK.

What about the spaces in the field names?

Unfortunately, Word for Windows won't be able to use the data file as it exists at this point. Word for Windows insists the field names that reside in the first paragraph contain no spaces. When you paste Access data in the document, the field names appear as they're spelled in the table definition. If the Access field names contain spaces, Word for Windows produces an error message as you try to attach the file as the mail merge data file.

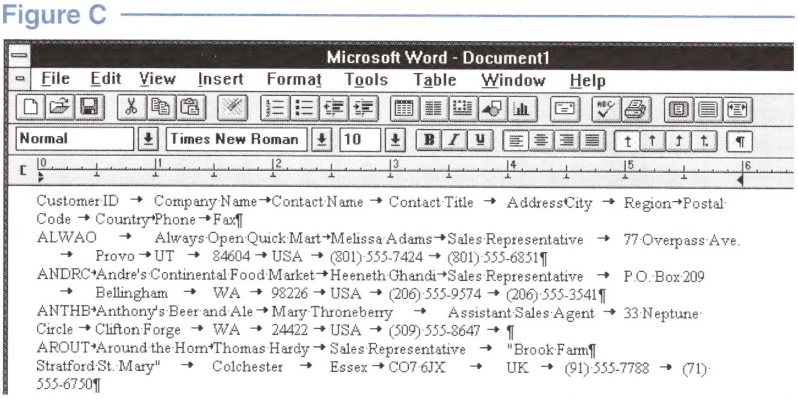
Figure C shows a magnified picture of the field headers in Figure B. Notice that the fields Customer ID, Company Name, Contact Name, and Postal Code contain spaces.

You must delete those spaces after you paste the data into the empty document and before you attach the file to a mail merge document. If you've upgraded to Version 1.1, you can use the new export option instead of cutting and pasting.

Attaching the data file

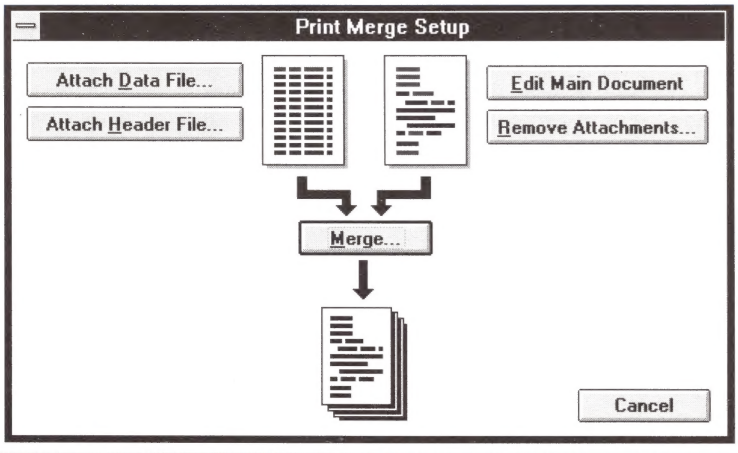
Once you remove the spaces from the field names, you can attach the data file to your letter document by using the File menu's Open... command.

Figure C



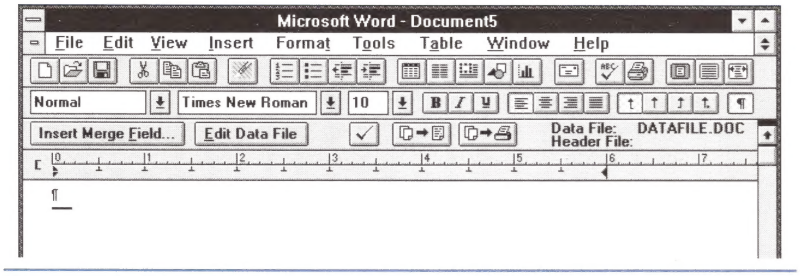
You should make sure the field names in the data file's header don't include spaces.

Figure D



You attach a data file to the letter document with the Print Merge Setup dialog box.

Figure E



When you attach the DATAFILE.DOC data file, Word for Windows adds a ribbon to the top of the window.

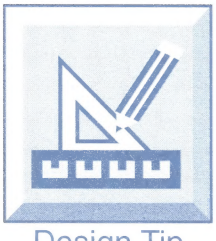
If you want to create a new document, use the File menu's New... command. Next, pull down the File menu again and select the Print Merge... command. Word for Windows will pop up the Print Merge Setup dialog box shown in Figure D.

To attach the data file, click the Attach Data File... button. Then, in the Attach Data File dialog box, select DATAFILE.DOC. When Word for Windows returns to the document, a new ribbon at the top of the window offers several buttons for creating and printing the mail merge document. Figure E shows the new ribbon just above the ruler.

Now you can type the letter and include the data file fields as you would normally. When you want to use data from the data file, click the Insert Merge Field... button.

Conclusion

In this article, we showed you how to create a Word for Windows data file by simply copying data from an Access table to the Clipboard and then pasting the data to a Word for Windows document. When you paste the data, the field header that Word for Windows requires in a data file automatically appears. Next month, we'll profile the new version's export feature, which provides an alternative method for filling a Word for Windows data file with Acess data. ◆



Design Tip

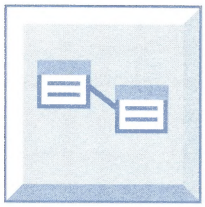
Sizing a label control to exactly the size of its contents

In researching the questions Robert Anderson raised in his letter last month ("Hints for Centering Labels on Forms and Reports"), we noticed an interesting trick in sizing a label control. We placed a label control on a new form, typed the text of the label, and then increased the font size from 8 to 18. Of course, the small size of the control hid most of the label text.

In a situation such as this, you'll want to increase the size of the control to show all of the label text. We've found that when you simply click inside the label control (as if you wanted to edit the text), Access automatically sizes the control to fit the new size of the text. Alternatively, you can pull down the Layout menu and select the Size To Fit option. Either way, the control will grow to fit the text.

You might think it would be just as easy to grab one of the corner handles and drag the mouse pointer until the control is the size you want. There's only one problem with that technique: If you've activated the Snap To Grid feature, the new size will snap to the grid. If you use either of the first two methods, Access will ignore the grid and fit the control around the text as closely as it possibly can. ◆

Understanding the outer join



When you join two tables in a query, you simply click the linking field in one table's field list and drag it to the linking field of the other table. When you do so, you create a type of join called an *equi-join*. When you create an equi-join, the query selects records only if the tables have matching entries in the join fields.

In some situations, the equi-join is too restrictive. You may want the query to select all records from a table and simply draw information from the other table for those records that have a linking entry. This type of link is called an *outer join*. To create this type of link, you first create an equi-join and then convert it to an outer join by using the Join Properties dialog box. In this article, we'll walk you through the steps for creating this special type of link.

The Join Properties dialog box

To convert an equi-join to an outer join, you double-click the join line that connects the tables in the Query window. Access pops up the Join Properties dialog box, which lets you choose how you want Access to select the records.

This dialog box provides a bank of three radio buttons. Option 1 selects the ordinary equi-join. (Obviously, it's the default option.) Options 2 and 3 define the two possible outer joins. Option 2 tells Access to include all records from one table; option 3 tells Access to include all the other table's records. Access customizes the options' descriptions to tell you how each option selects records.

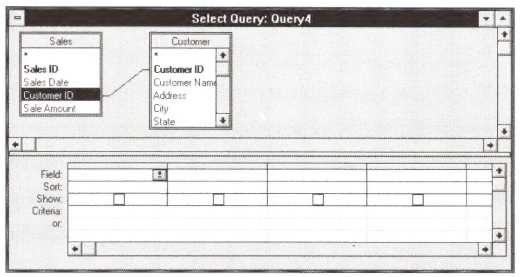
An example

To demonstrate how you use the Join Properties dialog box to create an outer join, we'll show you a common example. If you keep track of your company's sales with Access, you probably use separate tables to store sales information and customer information. However, you might create a query that combines the data from these tables in order to review the sales information. To relate these tables in a query, you'd probably use an outer join so you don't unknowingly omit sales records that are missing customer information.

Let's look at this situation more closely. Tables A and B show structures you might use for the Sales and Customer tables. As you can see, both tables have Counter key fields.

You relate the tables in a query by joining the tables' Customer ID fields. Figure A shows this type of query.

Figure A



To combine information from the Sales and Customer tables, you'd link the tables through the Customer ID field.

If you're confident that all Sales records have a valid Customer ID entry, using an equi-join to link the tables would be perfectly acceptable. However, in rare cases, Sales records may be missing their Customer ID entries or, perhaps, have an invalid entry. In these cases, the query would omit the Sales record from the datasheet.

In many situations, you might want to omit records that don't have a valid

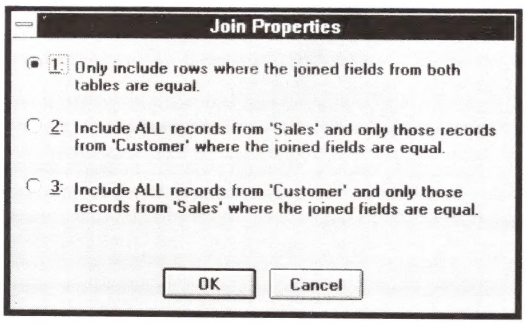
Table A

Key	Field Name	Data Type	Properties
🔑	Sale ID	Counter	
	Sale Date	Date/Time	
	Customer ID	Number	Field Size=Long Integer
	Sale Amount	Currency	

Table B

Key	Field Name	Data Type	Properties
🔑	Customer ID	Counter	
	Customer Name	Text	Field Size=40
	Address	Text	Field Size=60
	City	Text	Field Size=15
	State	Text	Field Size=15
	Zip Code	Text	Field Size=10
	Phone	Text	Field Size=24

Figure B

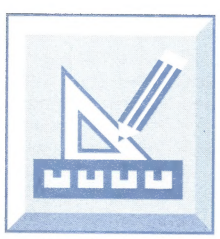


You use the Join Properties dialog box to convert the equi-join into an outer join.

relationship—but not in this case. You never want to leave out a Sales record. If Sales records drop out, you lose money. Therefore, you usually want to link the tables with an outer join that always includes Sales records and includes Customer data when it's available.

To create an outer join for this query, you double-click the join line that connects the Sales and Customer tables' Customer ID to invoke the Join Properties dialog box, shown in Figure B. As you can see, the descriptions to the right of the radio buttons explain each option's effect.

In this situation, you select option 2 in order to include all Sales records regardless of the contents of the Customer table's Customer ID field. When you click OK to return to the Query window, the join line will have an arrow pointing to the Customer table. This arrow signifies an outer join that includes all Sales records and looks up Customer data for Sales records that have a linking entry in the Customer ID field. ◆



Design Tip

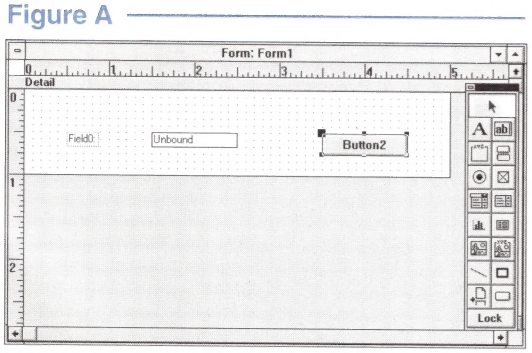
Defining access keys for controls on your forms

In the February/March issue of *Inside Microsoft Access*, "Open Access Objects in Design View by Using Keystroke Shortcuts" showed the significance of underlines in some of the menu options and control names. When you see an underline beneath a character, you can select that menu option or control by holding down the [Alt] key and pressing the character.

This type of key combination is called an access key, although the term *access* has nothing to do with the product Access. Almost all Microsoft products have access keys to certain menu commands and other operations.

Last month, Elisa Williams wrote to tell us those underlines were cropping up in her reports when she used the ampersand (&) in labels. As you may remember, we pointed out that an ampersand tells Access the character following it is the access key for that label control. Although we suggested you can create your own access keys in your forms, we didn't walk you through the process of doing so.

Figure A



We'll use the default text box and command button controls to demonstrate access keys.

This month, we'll show you how access keys can improve the way you work with a form's controls while you use the form. We'll start by creating a simple form on which you can experiment with access keys and show you a couple of examples.

An example form

Let's create a simple form that contains a text box and a button control. We'll assign these controls access keys and show you how easily you can access the controls by using the [Alt] key.

Start by clicking the Form button ( ) in the Database window and then clicking New. When the New Form dialog box appears, click Blank Form.

Now, place the controls on the form. Starting with the Text Box control, click the Text Box tool ( ) in the tool box, position the mouse pointer where you want the text box to reside, and then click about an inch and a half from the left border. Access will create the control and its associated label.

Next, place the button control by clicking the Command Button tool ( ) in the tool box and clicking an inch and a half from the right border. After you place both controls, your form should look something like the one shown in Figure A.

Defining an access key for the button

Now let's create an access key for the command button. You do this by selecting a letter for the access key from the button's Caption property and inserting an ampersand in front of it. Access will immediately place a thin line under the letter.

Let's use the *B* of the default Button2 Caption property as the access key. Open the Property sheet (if it isn't open already) by clicking the Properties button ( ) on the tool bar. Next, change the Caption property to *&Button2*. As soon as you move the cursor off that property, Access underlines the button caption's *B*.

Defining an access key for the text box

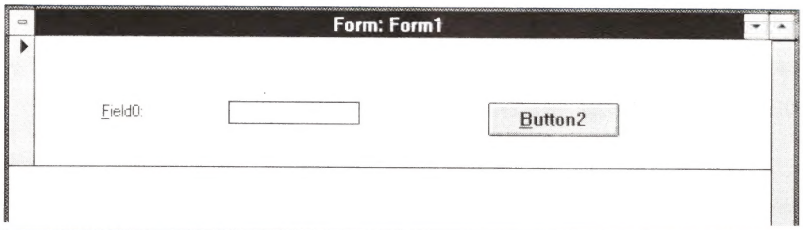
Creating an access key for the text box is a little trickier: You don't modify a text box property. Instead, you insert the ampersand in the Caption property of the text box's label control. The association between the text box and its label allows Access to move the cursor to the text box when you press the label's access key.

To see for yourself, click the text box's label and, in the property sheet, insert the ampersand in front of the *F* in the default Caption property Field0. Again, when you move the cursor off the Caption property, Access underlines the label's *F*.

Using the access keys

Now that you've defined the access keys for the controls, press the Form View button ( ) on the tool bar in order to test the access keys' operation. Figure B shows the example form. As you can see, the button caption's *B* and the field caption's *F* are underlined. Now try switching between the two controls by pressing their access keys. When you press [Alt]B, Access moves to the button control. When you then press [Alt]F, Access places the cursor in the text box control.

Figure B



The underlines in the controls' captions specify the access keys.

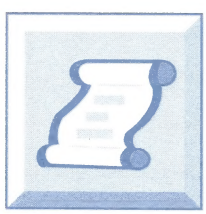
Defining access keys for other types of controls

You can define an access key for any type of control you can move to or select while working with a form. These controls include option groups, option buttons, check boxes, combo boxes, and list boxes. You define access keys for these controls the same way you do for text boxes. You just insert the ampersand in the Caption property of the control's associated label.

Conclusion

In this article, we showed you how to create access keys for the controls you place on your forms. Sometimes, using the keyboard to press access keys is an easier way to navigate the form than using the mouse. ◆

Disabling individual macro statements



Do you ever wish you could see how a macro works without one of its actions? You don't necessarily want to delete the action, because you may later decide to use it, but you want to see what happens if you skip the action.

If you have any programming experience, you're accustomed to disabling commands and statements by converting them into a comment. That way, the compiler will ignore the statements as it builds the executable program. Then you can test the result, and if you determine the program is better without the disabled lines, you can then delete them.

How you "comment out" a statement varies from language to language. For instance, you comment out a line in Access Basic by inserting an apostrophe (') at the beginning of the line, while in the C language, you enclose statements between the character pairs /* and */.

Unfortunately, you can't just comment out a macro action. Access provides a Comment column in which you can annotate the action without modifying the actual statement. However, there's another way to disable a macro action. In this article, we'll show you how, and we'll describe one situation in which this technique has come in handy for us.

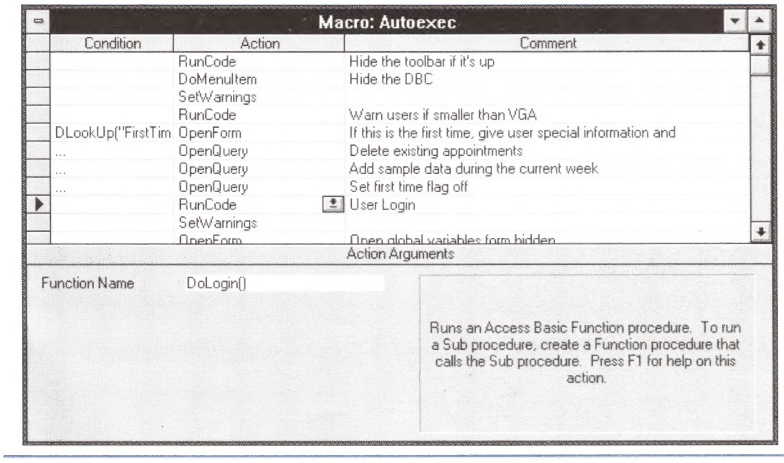
The technique

To comment out—or disable—a macro action, open the macro in Design view and open the Condition column by clicking the Conditions button ( ) on the tool bar. Then, in the Condition cell of the action you want to disable, you enter the conditional expression *1=2*.

Access will carry out the action only if the expression in the Condition column evaluates to a True value.

Now, why does this work? When Access tries to execute the action, it will first evaluate the conditional expression. Access will carry out the action only if the expression in the Condition column evaluates to a True value. Since 1 never equals 2, Access will always skip the action. (Of course, you could enter any conditional expression that always equals False.) This technique doesn't actually comment out the action. Instead, it simply prevents Access from executing the action.

Figure A



To try to remove the PIM application's Password dialog box, you must modify the database's Autoexec macro.

An example

We first used this technique while researching a problem for reader Robert Cote. Mr. Cote uses the PIM application Microsoft ships with Access to help manage his business appointments. He called us to ask if he could disable the password dialog box. He works in a single-user environment and doesn't want to bother with security.

We quickly learned that a single function called DoLogin( ), which the PIM developers created, controls user logon, and a RunCode action in the database's Autoexec macro calls DoLogin( ). We assumed that we could block the Password dialog box by removing the RunCode action. However, we didn't recommend that Mr. Cote delete RunCode, because the DoLogin( ) function may do other things than simply manage a dialog box. Therefore, we recommended he just temporarily disable the action. If the PIM application continued to work properly for a few days, he could then safely delete the line.

Let's look at the steps required to make this change. First, open the PIM database while pressing the [Shift] key. (If you don't hold down the [Shift] key, Access will automatically run the Autoexec macro.) Then, click the Macro button in the Database window and find Autoexec in the list of macros. Next, double-right-click the macro to open it in Design view. Access will produce the window shown in Figure A.

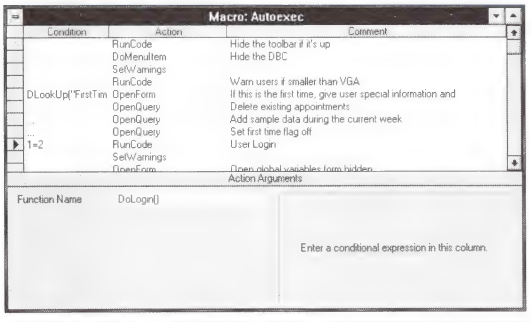
The ninth row executes the RunCode action that calls the DoLogin( ) function. To disable this action, first press the Conditions button on the tool bar and, in that row's Condition cell, type *1=2*, as shown in Figure B. Then, save the updated Autoexec macro and close the window. Also, close the Database window so you can reopen the database in order to test the modified Autoexec macro.

If you make this change to the database's Autoexec macro, the next time you open the PIM database, the Password dialog box won't appear, and the application will seem to work fine. When Mr. Cote tried it, he was happy—that simple change seemed to solve the problem.

Unfortunately, we soon found out that disabling the DoLogin( ) function call did *not* satisfactorily solve the problem. Mr. Cote called back the next day to report that the application no longer displayed the appointments for the current day. It displayed the

previous day's appointments—the day he disabled the macro action. Apparently, the DoLogin( ) function also picks up the current date that the Autoexec macro uses to determine which day's appointment to display.

Figure B



To omit the Password dialog box, disable the action that calls the DoLogin( ) function.

Since he only disabled the RunCode action in the macro, he could reinstate the application to its original operation by removing the *1=2* expression in the Condition cell.

Disabling several consecutive actions

The previous example illustrated a case in which you want to disable a *single* macro action. Occasionally, you may want to disable a block of consecutive actions. To do this, you simply apply the conditional expression to all the statements. You can bring consecutive actions under one conditional expression by entering ellipses in the Condition cells of the additional actions. Therefore, to disable consecutive actions, you enter the conditional expression *1=2* in the first action's Condition cell and ellipses in the additional cells.

To demonstrate this technique, suppose you want to disable the SetWarnings action that follows the RunCode action you disabled in the example. To do this, you'd enter an ellipsis in the row's Condition cell, as shown in Figure C.

When the statements already depend on conditions

The technique we've shown you works well when the action you want to disable doesn't already depend on a condition. If it depends on a condition, the technique will still work, but its implementation will be more complicated.

If the action is the only one governed by the condition, you simply insert the string *1=2 And* in front of the existing conditional expression. The extra condition will guarantee the expression will evaluate to False. To reinstate the original action, just remove *1=2 And*.

On the other hand, if the action you want to disable is part of a block of actions that already depend on a condition, implementing this technique might be too messy to be worthwhile. You'd have to copy the condition to each row rather than use the ellipses, and you'd insert the *1=2 And* conditional expression for the action you want to disable.

Notes

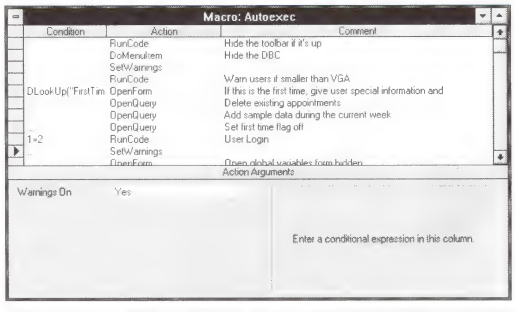
When you want to disable an action in a macro, you want to prevent Access from evaluating the row's conditional expression. As you may know, you can call Access Basic functions in the Condition column, which can perform operations as easily as the Action column can perform them.

By following our exact instructions and inserting the *1=2 And* expression before the existing conditional expression, you automatically suppress the existing condition. Why? Well, when Access evaluates a conditional expression, it doesn't do any more work than is absolutely necessary. The And keyword in a conditional expression tells Access that both pieces of the expression—the ones before and after the *And*—must evaluate to True. If the first piece of the expression evaluates to False, Access knows the full expression can't possibly evaluate to True. As a result, it doesn't bother to evaluate the rest of the expression.

Conclusion

In this article, we showed you an easy way to disable a macro action if you want to see how the macro performs without it. When you're designing a macro, this technique can be quite valuable because you can try various methods without erasing any of the trial methods. After experimenting for a while, you can choose the best method and then remove the rest. ◆

Figure C



To disable the action that follows the DoLogin( ) function call, use an ellipsis to continue the 1=2 *conditional expression.*

Form Tip

Creating identifiers that reference controls on subforms

When you design complex forms, reports, and macros, you often rely heavily on identifiers. You use identifiers to look up or change the values and properties of controls. You can even reference controls on forms other than the one you're working on.

Although the syntax of identifiers can look intimidating, the identifiers are fairly easy to build. However, there's one case in which creating an identifier is confusing—when you want to reference a control on a subform. In this article, we'll describe how you create identifiers for this purpose. But first, let's briefly review how to build basic identifiers.

Building identifiers

The simplest identifier you can create is one that references a control on the current form. You just enclose the control name in brackets. For instance, when you create the first text box control on a new form, its default name is Field2. The identifier that references the control is [Field2].

However, identifiers can be much more complicated. You can include solely the control name only when the control you want to reference is on the current form. If you use an identifier in a macro, you can use just the control name only if the control resides on the form from which you run the macro. When you want to reference a form control in a query, a report, or on a different form, you must provide information that tells Access how to find that control.

To convey that information, identifiers use several components, each separated by an exclamation point. The first part specifies the type of system object that holds the control you want to reference. You generally use the keyword Forms or Reports for this element. The second component indicates the name of the object that contains the control—in other words, the name of the form or report. Finally, the third part is the name of the actual control.

Let's look at an example. Last month's article "Using Query By Form to Replace a Parameter Query's Dialog Boxes" used the identifier

```
[Forms]![Customer Names From A State]!
➥ [Input State]
```

in a query's Criteria cell in order to refer to a form control. We used *Forms* first, since we were accessing a form control; we used *Customer Names From A State* as the second element to specify the form that contains the control; and we included *Input State* as the third component to specify the name of the control we wanted to reference.

Accessing control properties with identifiers

You can also use identifiers to access control properties. To do so, you append to the identifier the name of the property, separating the property name from the rest of the identifier with a period. For instance, in the previous example, you'd reference the Input State control's Visible property with the identifier

```
[Forms]![Customer Names From A State]!
➥ [Input State].Visible
```

Before moving on to our main topic, referring to controls on subforms, let's discuss the basic structure of the identifier. Notice that each component of the identifier is contained by the component to the left. For instance, the control is contained by the form it resides on. And when you think about it, the form is contained by the database, which the first part of the identifier—the system object—represents. As you'll see, this is an important way to look at identifier construction—especially when you try to make sense of the way you reference controls on subforms.

Referring to controls on subforms

When you refer to a control on a subform, you must insert a new component into the identifier. You place the name of the subform control immediately after the form name and before the name of the control you want to reference. This placement makes sense, since that position is the proper place in the hierarchy for a subform: The form contains the subform, which in turn contains the control.

However, the way the new component fits into the expression isn't quite intuitive. You don't simply insert the object's name between exclamation points. You must also tell Access the new component identifies a subform

rather than a simple control such as a text box. You do so by appending a period and the keyword Form to the name of the subform control. For instance, the identifier

```
[Forms]![Form1]![Subform Control Name].Form!
➥ [Field1]
```

refers to the control named Field1 on the form named Subform that's embedded on the form Form1.

Note that the subform control name is *not* the same as the subform's name that appears in the Database window. Remember, the subform control is simply a control that displays the subform. The subform control has its own name, but the name can often be the same as the subform's name. However, when it's different, you must use the subform control name in the identifier.

An example

Let's work through a simple example. You'll create a blank form and embed the NWIND database's Products form on it. Then, you'll create a new text box control on the blank form that displays a value in a control on the Products subform.

Open the NWIND database by using the File menu's Open Database... command. Next, click the Form button in the Database window and then click the New button. In the New Form dialog box, click the Blank Form button.

Now embed the Products form as a subform. Return to the Database window and scroll through the list of forms until you find Products. Then, drag and drop the form onto the blank form you just created. Next, return to the blank form and reposition the subform control to the left side of the form and about a quarter inch from the top. You may need to resize the Form window to fit the entire subform control on the screen. Figure A shows the new form at this point.

Next, place a text box control in the area above the subform, as shown in Figure B. Then, open the property sheet by clicking the Properties button ( ) on the tool bar and change the new text box control's Control Source property to

```
=[Products].Form![Product Name]
```

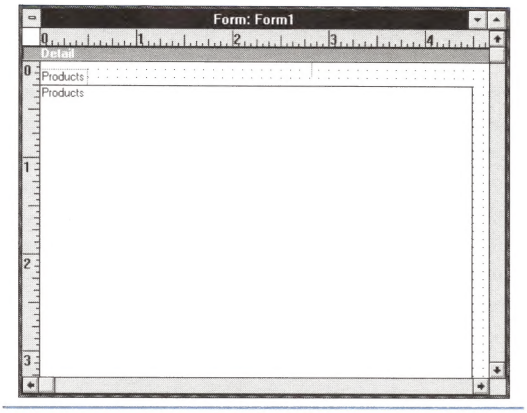
which tells Access to look up the value of the Product Name control on the Products subform control. In this particular case, we don't need to include the Form system object or the name of the form, because the form is the current object.

Now when you view the form by clicking the Form view button on the tool bar, the new text box will display the value of the subform's Product Name control, as shown in Figure C. You can click the subform's navigation buttons to change the current record on the subform and watch the text box value change to match the record in the subform.

Conclusion

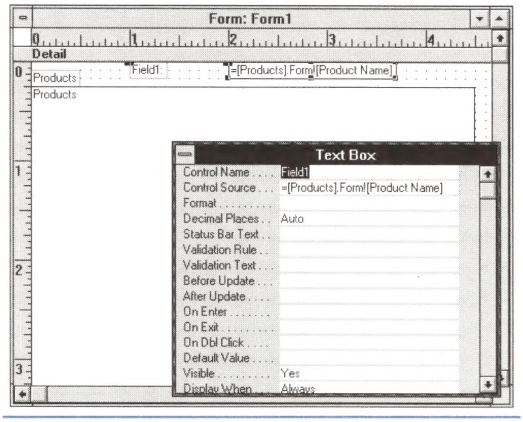
In this article, we reviewed how to use identifiers to reference controls on forms. Then, we showed you the undocumented way to reference controls on subforms. The trick is to use the subform control's name rather than the form name as it appears in the Database window. Then, you append the subform control name with a period and the keyword Form. By doing so, you identify the component of the identifier as a form rather than an ordinary control. ◆

Figure A



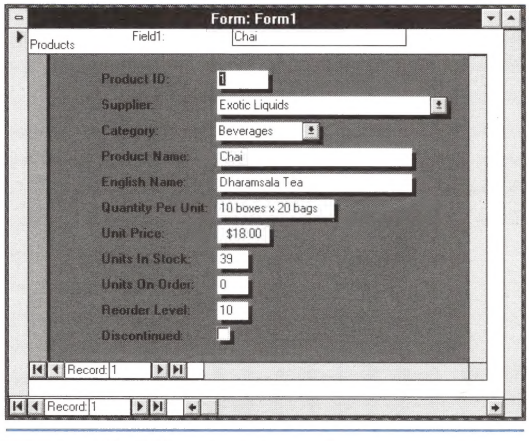
Drag and drop the Products form onto a new form.

Figure B

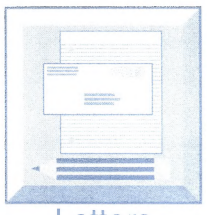


Place a text box control on the main form to display the value of a control on the Products form.

Figure C



Using an identifier as the control source, you can make the text box display field data from the Products form.



Letters

Selecting records from one table that aren't in another table

I have a table named Mailing List that stores names and addresses of people to whom I sent a direct-mail offer. When I received responses, I entered them into another table named Responses. Figure A shows some sample data in those two tables. Note that both tables use the Contact Name field as their key field.

Now that I've received all the responses I expect to get, I want to analyze the effectiveness of the offer. I want to select from the Mailing List table all the names that don't appear in the Responses table. Then, I can send a follow-up ad to those people.

I can easily create a query that lists all the people who responded. I simply join the Mailing List and Responses tables by the Contact Name field. The resulting query selects the records that have matching entries in the Contact Name field. However, Access doesn't seem to provide a way to select the records from the Mailing List table that don't appear in the Responses table. Do you have any suggestions?

Stephen E. Einson
New York, New York

Identifying records that two tables don't have in common can be a problem. As Mr. Einson says, simply joining the tables in a select query returns the names you don't want to see—those the tables *do* have in common.

Fortunately, you can modify the simple select query to get the results you want. This special type of query is called a *not-in query* because it selects the records in one table that are "not in" another.

Figure A

Table: Mailing List

Contact Name	Contact Title	Address	City	State	Postal Code	Phone
Art Braunschweiger	Sales Manager	P.O. Box 555	Lander	WY	82520	(307) 555-4680
Bill Blandings	Sales Manager	234 Samuel Pl.	Ithaca	NY	14853	(607) 555-9679
Bill Lee	Assistant Sales Representative	418 Datability Ave.	Pocatello	ID	83201	(208) 555-9787
Chiara Santorini	Marketing Assistant	45 N. Terminal Way	Helena	MT	59601	(406) 555-9725
David Oberholter	Sales Associate	340 Mile High Blvd.	Denver	CO	80202	(303) 555-6535
Grover Smith	Owner	418 - 6th Ave.	Walla Walla	WA	99362	(509) 555-7689
Gyorgy Gheorghu	Sales Manager	2782 - 92nd Ave.	Los Angeles	CA	90071	(213) 555-6360
Hank Proudlit	Accounting Manager	45 E. 23rd St.	Drain	OR	97435	(503) 555-5840
John Steel	Marketing Manager	12 Orchestra Terrace	Walla Walla	WA	99362	(509) 555-7969
Jose Pavarotti	Sales Representative	187 Suffolk Ln.	Boise	ID	83720	(208) 555-8097
Judy Pamona	Owner	89 Rain Way	Portland	OR	97219	(503) 555-9753
Karl Jablonski	Owner	1029 - 12th Ave. S.	Seattle	WA	98124	(206) 555-4112
Liz Nixon	Marketing Manager	89 Jefferson Way	Portland	OR	97201	(503) 555-3612
Louisa Scarpaczyk	Sales Representative	1225 Zephyrus Rd.	Anacortes	WA	98221	(206) 555-8647
Margaret Morgan	Marketing Manager	33 Upper Arctic Dr.	Buffalo	NY	14240	(716) 555-5686
Norman Jones	Sales Representative	361 Pitt St.	Jamaica	NY	11451	(607) 555-9947
Olivia LaMont	Marketing Manager	3086 Oakridge Ave.	San Francisco	CA	94965	(415) 555-6940
Rona Rumalski	Marketing Manager	Evans Plaza	Eugene	OR	97403	(503) 555-5831
Sandy Lapworth	Sales Associate	P.O. Box 10029	Austin	TX	78759	(512) 555-4524
Sean O'Brien	Assistant Sales Agent	98 N. Hyde Dr.	San Francisco	CA	94103	(415) 555-7357
Shannon MacArthur	Marketing Assistant	Eastgate Center	Bellevue	WA	98006	(206) 555-9575

Record: 1

Table: Responses

Contact Name
Art Braunschweiger
Bill Blandings
Bill Lee
Chiara Santorini
David Oberholter
Grover Smith
Gyorgy Gheorghu
Hank Proudlit
John Steel
Jose Pavarotti
Judy Pamona
Karl Jablonski
Liz Nixon
Louisa Scarpaczyk

Record: 1

The Mailing List table stores names of people who receive a direct-mail offer; the Responses table holds the names of those who responded.

Building a not-in query involves some advanced query techniques. You must understand how to create an *outer join* rather than the usual type of link Access creates, called an *equi-join*. You must also know how to select records based on summary criteria. You may want to read the articles beginning on page 1 ("Selecting Records Based on Summary Information") and on page 7 ("Understanding the Outer Join") for explanations of these techniques.

The not-in query

Before we show you the detailed solution to Mr. Einson's problem, we'll provide a quick summary of the general technique. You follow these basic steps:

- Create a query with an equi-join that selects records only when the tables involved have matching records.
- Convert the equi-join to an outer join so that the query will select all records in the table with the larger number of records—regardless of whether the other table has a matching record.
- Define query criteria on the count of the smaller table's linking field, letting the query select records only if the count equals 0.

In summary, the outer join you create in step 2 prevents Access from eliminating the records that the tables don't have in common. Then, the criteria you set in step 3 identify records in the one table that aren't in—or have a count of 0 in—the other table.

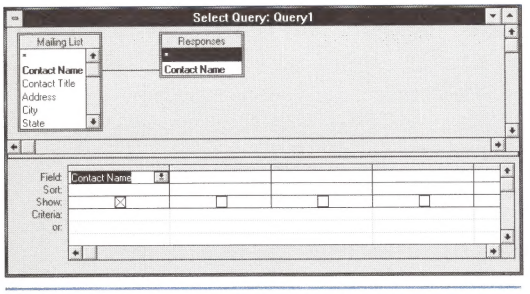
An example

Now let's apply these steps to build the not-in query Mr. Einson needs. Start with the simple select query that links the Mailing List and Responses tables with an ordinary equi-join. To create this query, set up a query that involves the two tables. To do this, move to

the Database window and highlight the Mailing List table. Then, click the New Query button ( ) on the tool bar. Next, press [F11] to return to the Database window and then drag the Responses table into the Query window.

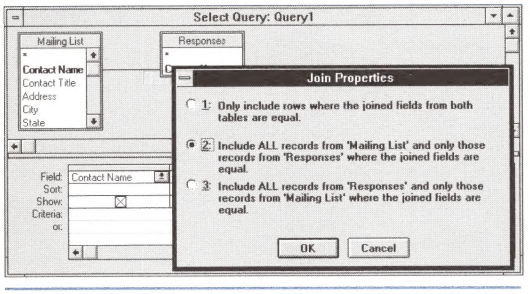
Next, create the basic query that selects names the tables have in common. Do this by clicking Contact Name in the Mailing List table's field list and dragging the mouse pointer to the same field in the Responses table's field list. Then, drag the Mailing List table's Contact Name field to the QBE grid. Figure B shows this basic query.

Figure B



The select query above returns the records in the Mailing List table that have a matching record in the Responses table.

Figure C



You choose the type of join you want by using the Join Properties dialog box.

Now, convert the standard equi-join to an outer join. Open the Join Properties dialog box by double-clicking on the join line; then, select option 2 and click OK. Figure C shows the Join Properties dialog box for this example.

Next, you drag the Contact Name field from the Responses table to the second column of the QBE grid and deselect the show box. Then, click the Totals button ( ) on the tool bar. In the new column's Total cell, click the dropdown arrow and choose Count from the selection list. Finally, enter *0* in the column's Criteria cell. Figure D shows the completed query. (Notice that we've opened the Table row to show you the tables each field belongs to. You do this by pulling down the View menu and selecting Table Names.)

When you now press the Datasheet View button ( ) on the tool bar, the query will select the Contact Name entries in the Mailing List table that don't have corresponding entries in the Responses table. Figure E shows the records the query selects when the tables store the data shown in Figure A.

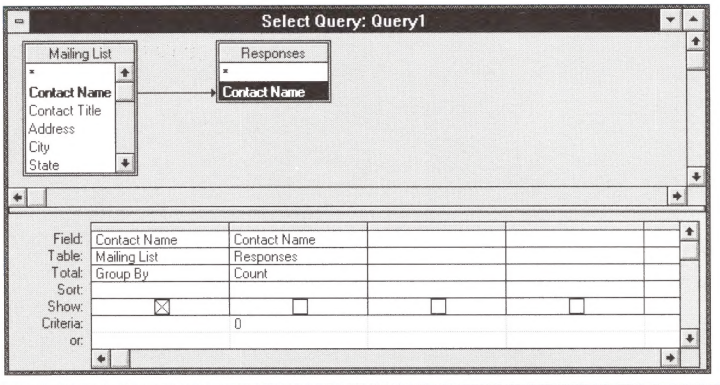
If you compare the data in these figures, you'll see that the query does indeed select the records in the Mailing List table that aren't in the Responses table.

Notes

The example query we showed you displays only one field from a table. If you want, you can include other fields by simply dragging them to the QBE grid.

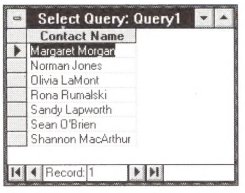
If you read "Selecting Records Based on Summary Information," on page 1, you may be wondering how we can suggest this. In that article, we said you can't add fields to a query that places criteria on summary calculations, because doing so changes how the query will group the data in preparation for the summary operation. Well, in this case, the Group By operator in the Total cell of the key field already tells Access to group by the individual records. ◆

Figure D



This not-in query will select records from the Mailing List table that aren't in the Responses table.

Figure E



Our sample query selects these contact names from the data shown in Figure A.

SECOND CLASS MAIL

Microsoft Access
Technical Support
(206) 635-7050

Please include account number from label with any correspondence.

Can you use the Format property to set a counter field's starting value?

I received the May issue of *Inside Microsoft Access* today and was surprised to find your article "Initialize Counter Fields with a Simple Append Query," which described how to set a counter field's starting value. I use a much simpler method than the one you describe. I use the counter field's Format property to tell Access the value to display.

You should keep in mind that the Format property affects only the counter field's display.

To do this, you use a backslash followed by the number on which you want to base the counter values. For instance, if you want Access to number records beginning with 1001, you open the table in Design view, highlight the counter field, and assign to the field's Format property the entry *\1000*.

Kevin M. Jividen
Cuyahoga Falls, Ohio

Mr. Jividen is correct; you *can* set a counter field's starting point by using the field's Format property. However, you should keep in mind that the Format property affects only the counter field's display. It does *not* affect counter values Access generates and stores for new records.

To see evidence of this fact, you can try to join a formatted counter field to match a counter field in another table. If you join the tables in a query by the counter field they have in common, Access won't consider the special formatting when associating the records. For instance, if you use the formatting code *\10* for one table's counter field, Access will join record 11 with record 1 in the other table, 12 with 2, and so on. This can be very confusing.

On the other hand, if both tables use the *\10* formatting code for their counter fields, Access will link the records just as you'd expect. In short, as long as you keep formatting consistent for counter fields that relate to fields in other tables, you can use the Format property in order to display counter values at whatever starting point you choose.

Potentially, there's a more serious problem with using the Format property to set a counter field's starting value. Access doesn't actually use the number in the formatting code as an offset, or starting point. For instance, suppose you use the format code *\10* for a table's counter field and the last record has the value 19 in that field. If Access used the formatting code number as an offset, you'd expect Access to display 20 in the next record's counter field. However, Access displays 110 instead.

Apparently, in formatting the counter value, Access replaces the zeros in the formatting code with the counter value. When the counter value contains more digits than there are zeros in the formatting code, Access forces the counter value with the extra digit in place of the zeros. In the previous example, Access replaced the single zero of the formatting code *\10* with the new counter value, 10. The display value was therefore 110.

As a result, the number you choose for the formatting code should be larger than the number of records you expect to add to the table. When the counter field values reach that limit, the effectiveness of the formatting code breaks down. ◆

Printed in the USA
Printed on recycled and recyclable paper
50% recycled fiber including 10% post-consumer waste